An Ada Interface to a Functionally Designed Library

by Jeffrey H. Simonson

Interfacing Ada to other languages sometimes involves not only data flow interfaces, but also design methodology interfaces. This article describes an object-oriented to function-oriented interface to the DEC VAX Screen ManaGement (SMG) Run-time Library facility.

This article was originally published in *Ada Rendezvous* Vol. VI, No. 1, Texas Instruments, Inc., 1989.

Introduction

Various design methods are used to manage the complexities of software systems. With the development of Ada, interest in object-oriented design has increased. Although not a true object-oriented language, such as Smalltalk, Ada has features which encourage the real world modeling technique. Unfortunately, other previously written systems which the object-oriented Ada programmer may want to use might not have been designed with the same method. Such is the case with the VAX Screen ManaGement facility (SMG).

This article discusses the general method of object-oriented design and uses an interface to SMG as an example. Since SMG is fundamentally function-oriented, this will require not only interface of languages and data flow, but development methods as well. This strategy will both capture real world object relations and insulate the Ada application from unwanted SMG details.

Object-Oriented Development Method

Capturing abstractions of real world entities and their relations is fundamental to object-oriented design. It emphasizes the importance of software objects and the operations each object can perform. In contrast, the conventional methods of top-down structured design and data-structure design lose either the object abstractions or the object relationships.

Several approaches to object-oriented design exist. This interface design applies the method proposed by Grady Booch. His approach includes the following steps:

1. **Identify the Objects** - Determine the major actors of the problem space. They are entities which have a value and are usually derived from noun phrases.
2. **Identify the Operations** - Determine the operations that can be performed on each object. These will be the procedure and functions of the program.
3. **Establish the Visibility** - Identify the dependencies between objects. This is where actor relationships and visibilities are defined.
4. **Establish the Interface** - Produce a formal Ada package specification to capture the relations established in the previous step. This forms a boundary between the inside and the outside view of an object.
5. **Implement each Object** - Compose the operations in the routine bodies. This may involve more decomposition by repeating the development method.

SMG Overview

The VAX Screen Management facility provides terminal independence for screen functions. Software emulation is provided for terminal hardware which does not support a particular function (for example, scrolling a sub-region of the screen). These procedures are primarily intended for video displays, but they can also be used with hardcopy devices and files.

The real power of SMG, however, comes from its complex image composition support. This feature is carried out using virtual displays and keyboards. A virtual display can define any part of the physical terminal, but application routines see the terminal in terms of a particular virtual display definition; this separates user programs entirely from the physical device.

Interface Goals

The goal of this Ada interface to the SMG library has two major objectives. First, it must, of course, be functionally equivalent to SMG. Second, the solution must map directly to the problem space and preserve the real world view of SMG objects throughout the design and implementation phases. Booch's design approach is used in this implementation to achieve an interface from the functionally-designed library to an object-oriented application. The interface will hide details not needed by the user and emphasize SMG objects and their relations.

Interface Implementation

The Ada interface is implemented by simply following each step of the design method described above. The first task is identification of the objects; these will be the package names. Major actors in SMG are:

	Physical pasteboards - These map directly to a physical device.
	Virtual displays - These are created and pasted to a pasteboard.
	Cursors - These are place holders normally associated with displays.
	Characters - These are printed on a display at the cursor position.
	Lines - These are composed of characters.

	Virtual keyboards - These are simply input devices.
	Other minor objects - These are bells, broadcast messages, rectangles, etc.

Next, the operations are defined to create the Ada routine names. This simply reflects the verb functions available in SMG. For example, pasteboards can be *repainted*, displays *scrolled*, cursors *positioned*, characters *put* on displays, lines *drawn*, and keystrokes *read*.

Step three involves establishing the relationships of each object to other objects. Conventional design methods tend to lose these relationships; this interface strives to recapture them. The major actors are related in a simple hierarchy. Virtual displays are pasted on pasteboards; cursors, characters, and lines are printed on virtual displays; and keyboards are tied to pasteboards. Other minor relationships also exist.

In step four, sub-packages are used as a representation of actual object relationships. Since virtual displays are pasted on pasteboards, Package DISPLAY resides within Package PASTEBOARD. A similar structure appears within Package DISPLAY which contains the objects *CURSOR*, *CHARACTE* , and *LINE*. Figure 1 shows this package structure.

```
package SCREEN_MANAGER
  generic package PASTEBOARD

    generic package KEYBOARD
    end KEYBOARD;

    generic package DISPLAY
      package CURSOR
      end CURSOR;
      package CHAR
      end CHAR;
      package LINE
      end LINE;
      end DISPLAY;

    end PASTEBOARD;
end SCREEN_MANAGER;
```

Figure 1: **Interface Package Structure**

To handle the overhead of creating and pasting, PASTEBOARD, DISPLAY, and KEYBOARD are generic packages which have as their parameters all the information necessary to do the creation overhead at elaboration time. Since each generic package instantiation renames the package, careful choices are necessary for creating meaningful Ada statements. For example, selecting CRT for PASTEBOARD, TOP for DISPLAY, and renaming SCREEN_MANAGER to SMG gives:

```
THE_ROW := SMG.CRT.TOP.CURSOR.ROW;
```

Now, procedures and functions (sometimes called methods of an object) are declared for each object in their respective package. All other information will be hidden and handled inside the package body. For example, Figure 2 shows a procedure as it appears in the SMG Run-Time Library. Figure 3 shows the same procedure as it appears in the interface package CURSOR.

```
function SMG$_CURSOR_POS (STATUS     : out INTEGER;
                          DISPLAY_ID : in  INTEGER;
                          ROW        : out INTEGER;
                          COLUMN     : out INTEGER);
```

Figure 2: **SMG Run-Time Library Function GET_CURSOR_POS**

```
procedure POSITION (THE_ROW    : out ROWS;
                    THE_COLUMN : out COLUMNS);
```

Figure 3: **Procedure POSITION in Package CURSOR**

The last step, implementing each object, merely involves filling in the bodies. The appropriate SMG library routine is called to cause the indicated function. Any information not in the ADA interface routine parameters can be found either in the generic package parameters or in static, package body objects. Output data, such as the status, can be handled internally.

Summary

To prevent the limitations of other packages from propagating into an object-oriented system, design methods and functionality should be considered when creating an interface. The same principles used to create an application can be applied to interface implementation. This allows the user to view a library facility with real world abstractions absent of unnecessary details.

Bibliography

1. *Software Engineering with Ada, Second Edition.* Booch, Grady. Menlo Park, California: Benjamin/Cummings, 1986.
2. *What's in an Object?* Thomas, Dave. BYTE, March 1989, pp. 231-240.
3. *VAX/VMS Run-Time Library Manual.* Digital Equipment Corporation.